Functional Hierarchy of Roles for the Sole Power Mobile App

This document outlines the functional hierarchy of roles within the Sole Power mobile app, as defined by the RFP. Each role will have its own set of functionalities and access levels.

1. Participant Role

- **1.1. Account Management**
  - **1.1.1. Sign Up:**
    - **Description:** Allow users to create a new account within the app.
    - **Sub-Tasks:**
      - **1.1.1.1. User Input:** Provide input fields for username, email address, and password. These fields should be clearly labeled and intuitive for users to understand. The username field should enforce a minimum and maximum length, allowing only alphanumeric characters and underscores. The email address field should validate the input using a regular expression to ensure a valid email format. The password field should enforce a minimum length and complexity requirements, including the use of uppercase and lowercase letters, numbers, and special characters.
      - **1.1.1.2. Validation:** Validate user input to ensure:
        - **Username Uniqueness:** The system should check if the entered username already exists in the database. If it does, the user should be prompted to choose a different username.
        - **Email Address Validity:** The system should validate the email address using a regular expression to ensure a valid format. This helps prevent typos and ensures that the email address is deliverable.
        - **Password Complexity:** The system should enforce a minimum length and complexity requirements for the password. This helps to ensure that the password is strong and difficult to guess.
      - **1.1.1.3. Social Media Integration:** Optionally allow users to sign up using existing social media accounts (e.g., Facebook, Google). This provides a convenient and streamlined sign-up process for users who already have social media accounts. The system should integrate with the relevant social media APIs to securely retrieve user information and create an account.
      - **1.1.1.4. Account Creation:** Create a new user account in the database with the provided information. This involves storing the user's username, email address, password (hashed for security), and other relevant profile information in the database.
      - **1.1.1.5. Email Verification:** Send a verification email to the user's provided email address. This email should contain a unique verification link that the user needs to click to activate their account. This helps to prevent spam and ensures that the email address is valid.
      - **1.1.1.6. Account Activation:** Activate the user account upon successful email verification. Once the user clicks the verification link in the email, the system should update the user's account status to "active" in the database. This allows the user to log in and access the app's features.
  - **1.1.2. Login:**
    - **Description:** Allow existing users to log in to their accounts.
    - **Sub-Tasks:**
      - **1.1.2.1. User Input:** Provide input fields for username/email and password. These fields should be clearly labeled and intuitive for users to understand. The system should allow users to log in using either their username or email address.
      - **1.1.2.2. Authentication:** Verify user credentials against the database. The system should compare the entered username/email and password with the corresponding information stored in the database. If the credentials match, the user is authenticated.
      - **1.1.2.3. Two-Factor Authentication:** Implement two-factor authentication (e.g., SMS code, authentication app) for enhanced security. This adds an extra layer of security by requiring users to provide a second factor of authentication in addition to their password. This can be implemented using SMS codes, authentication apps, or other methods.
      - **1.1.2.4. Session Management:** Create a secure session for the logged-in user. This involves generating a unique session ID and storing it in a secure cookie or local storage on the user's device. This session ID is used to identify the user during subsequent requests to the server.
      - **1.1.2.5. Password Reset:** Allow users to reset their password via email. If a user forgets their password, they should be able to request a password reset. The system should send an email to the user's registered email address with a link to reset their password. This link should expire after a certain time period to prevent unauthorized access.
  - **1.1.3. Profile Management:**
    - **Description:** Allow users to update their profile information.
    - **Sub-Tasks:**
      - **1.1.3.1. User Input:** Provide input fields for name, email address, password, and other relevant profile information. These fields should be clearly labeled and intuitive for users to understand. The system should allow users to update their name, email address, password, and other relevant profile information.
      - **1.1.3.2. Validation:** Validate user input to ensure data integrity. This involves checking if the entered data is valid and meets the required format. For example, the email address should be validated using a regular expression to ensure a valid format.
      - **1.1.3.3. Profile Picture Upload:** Allow users to upload a profile picture. The system should allow users to upload a profile picture from their device or choose a picture from their social media accounts. The system should also handle image resizing and optimization to ensure that the uploaded image is displayed correctly in the app.

        - **1.1.3.4. Social Media Linking:** Optionally allow users to link their social media accounts to their profile. This allows users to easily share their progress and achievements on social media. The system should integrate with the relevant social media APIs to securely retrieve user information and link their accounts.
        - **1.1.3.5. Data Update:** Update the user's profile information in the database. Once the user has entered and validated their profile information, the system should update the corresponding information in the database. This ensures that the user's profile information is up-to-date and accurate.
    - **1.1.4. Logout:**
        - **Description:** Allow users to securely log out of their accounts.
        - **Sub-Tasks:**
            - **1.1.4.1. Session Invalidation:** Invalidate the user's current session. This involves deleting the session ID from the user's device and the server. This prevents unauthorized access to the user's account after they have logged out.
            - **1.1.4.2. Clear User Data:** Clear any sensitive user data from the device's memory. This involves deleting any sensitive user data that is stored locally on the user's device, such as login credentials, trip data, and other personal information.
            - **1.1.4.3. Redirect to Login:** Redirect the user to the login screen. Once the user has logged out, the system should redirect them to the login screen. This ensures that the user cannot access the app's features without logging in again.
- **1.2. Trip Tracking and Logging**
    - **1.2.1. Trip Tracking (GPS):**
        - **Description:** Enable users to track their trips using GPS.
        - **Sub-Tasks:**
            - **1.2.1.1. Start Trip:** Allow users to initiate trip tracking. The system should provide a clear button or icon that users can tap to start tracking their trip. This should prompt the user to grant the app permission to access their location data.
            - **1.2.1.2. Location Tracking:** Continuously track the user's location using GPS. The system should use the device's GPS sensor to track the user's location at regular intervals. This data should be stored in the database for later analysis and visualization.
            - **1.2.1.3. Distance Calculation:** Calculate the distance traveled during the trip. The system should use the tracked location data to calculate the total distance traveled during the trip. This can be done using the Haversine formula or other distance calculation algorithms.
            - **1.2.1.4. Trip Details:** Record trip details: start time, end time, distance, mode of transportation, and trip category. The system should record the start time and end time of the trip, the total distance traveled, the mode of transportation used (e.g., walk, bike, run, e-bike, skate, other), and the category of the trip (e.g., work, social, errands).
            - **1.2.1.5. Map Display:** Display the user's current location and route on a map. The system should integrate with a mapping service (e.g., Google Maps, Apple Maps) to display the user's current location and the route they have traveled during the trip. This should be displayed in a user-friendly format with clear markers and labels.
            - **1.2.1.6. Pause/Resume Tracking:** Allow users to pause and resume trip tracking. The system should provide a button or icon that users can tap to pause or resume trip tracking. This allows users to temporarily stop tracking their trip if needed, for example, if they need to take a break or make a stop.
            - **1.2.1.7. End Trip:** Allow users to end trip tracking and save the trip data. The system should provide a button or icon that users can tap to end trip tracking. This should prompt the user to confirm that they want to end the trip and save the data. Once the trip is ended, the system should store the trip data in the database.
    - **1.2.2. Manual Trip Entry:**
        - **Description:** Allow users to manually enter trip details if GPS tracking is unavailable or not desired.
        - **Sub-Tasks:**
            - **1.2.2.1. User Input:** Provide input fields for date, time, distance, mode of transportation, and trip category. These fields should be clearly labeled and intuitive for users to understand. The system should allow users to enter the date, time, distance, mode of transportation, and category of the trip manually.
            - **1.2.2.2. Validation:** Validate user input to ensure data integrity. This involves checking if the entered data is valid and meets the required format. For example, the distance should be a positive number, and the mode of transportation should be selected from a predefined list.
            - **1.2.2.3. Trip Data Storage:** Store the manually entered trip data in the database. Once the user has entered and validated their trip data, the system should store it in the database. This ensures that the trip data is saved and can be accessed later.
    - **1.2.3. Trip Log:**
        - **Description:** Display a chronological list of all logged trips.
        - **Sub-Tasks:**
            - **1.2.3.1. Data Retrieval:** Retrieve trip data from the database. The system should retrieve the user's trip data from the database and display it in a chronological order.
            - **1.2.3.2. Trip Display:** Display trip details: date, time, distance, mode of transportation, and trip category. The system should display the trip details in a user-friendly format, with clear labels and icons.
            - **1.2.3.3. Filtering:** Allow users to filter trips by date, mode of transportation, and category. The system should provide filtering options to allow users to easily find specific trips. For example, users can filter trips by date range, mode of transportation, or trip category.
            - **1.2.3.4. Editing:** Allow users to edit trip details. The system should allow users to edit the details of their trips, such as the date, time, distance, mode of transportation, and category. This allows users to correct any errors or update their trip data.
            - **1.2.3.5. Deletion:** Allow users to delete trips. The system should allow users to delete trips from their trip log. This

allows users to remove unwanted or outdated trips from their data.
            - **1.2.3.6. Data Export:** Allow users to export trip data in a desired format (e.g., CSV, PDF). The system should allow users to export their trip data in a format that they can easily use for other purposes, such as sharing with others or importing into a spreadsheet.
- **1.3. Data Visualization and Insights**
    - **1.3.1. Personal Dashboard:**
        - **Description:** Display cumulative data for the current season and over time.
        - **Sub-Tasks:**
            - **1.3.1.1. Data Aggregation:** Aggregate trip data for the current season and over time. The system should aggregate the user's trip data for the current season (Memorial Day - Columbus Day) and over time (life of account to date). This involves calculating the total number of trips, total mileage, gas saved, CO2e reduced, and money saved.
            - **1.3.1.2. Data Visualization:** Display aggregated data in a user-friendly format (e.g., charts, graphs). The system should display the aggregated data in a visually appealing and informative format, such as charts, graphs, or tables. This helps users to easily understand their progress and achievements.
            - **1.3.1.3. Data Display:** Display metrics: number of trips, total mileage, gas saved, CO2e reduced, and money saved. The system should display the following metrics on the dashboard:
                - **Number of Trips:** The total number of trips logged by the user.
                - **Total Mileage:** The total distance traveled by the user.
                - **Gas Saved:** The estimated amount of gas saved by the user by choosing alternative modes of transportation.
                - **CO2e Reduced:** The estimated amount of carbon dioxide emissions reduced by the user by choosing alternative modes of transportation.
                - **Money Saved:** The estimated amount of money saved by the user by choosing alternative modes of transportation.
            - **1.3.1.4. Data Filtering:** Allow users to switch between viewing data for the current season, calendar year, and life of account. The system should provide filtering options to allow users to view their data for different time periods. This allows users to track their progress over time and compare their performance across different seasons.
    - **1.3.2. Leaderboards:**
        - **Description:** Display individual and team leaderboards for the current season and historical data.
        - **Sub-Tasks:**
            - **1.3.2.1. Data Retrieval:** Retrieve leaderboard data from the database. The system should retrieve the leaderboard data from the database, including the rankings of individual participants and teams for the current season and historical data.
            - **1.3.2.2. Leaderboard Display:** Display leaderboard data in a user-friendly format. The system should display the leaderboard data in a visually appealing and informative format, such as a list or table. This should include the user's rank, username, team (if applicable), and the relevant metrics, such as total mileage, gas saved, CO2e reduced, and money saved.
            - **1.3.2.3. Data Filtering:** Allow users to filter leaderboards by season and category. The system should provide filtering options to allow users to view leaderboards for different seasons and categories. This allows users to compare their performance with others in different time periods and categories.
            - **1.3.2.4. Historical Data:** Provide access to historical leaderboard data from previous years. The system should provide access to historical leaderboard data from previous years. This allows users to track their progress over time and see how their performance has improved.
- **1.4. Social Features**
    - **1.4.1. Team Management:**
        - **Description:** Allow users to create or join teams.
        - **Sub-Tasks:**
            - **1.4.1.1. Team Creation:** Allow users to create a new team with a name and description. The system should provide a form for users to enter the team name and description. The team name should be unique and should not already exist in the database.
            - **1.4.1.2. Team Joining:** Allow users to join existing teams. The system should display a list of existing teams that users can join. Users should be able to search for teams by name or description.
            - **1.4.1.3. Team Member Management:** Allow users to view team members and their progress. The system should display a list of team members, including their usernames, profile pictures, and their current progress in the challenge.
            - **1.4.1.4. Team Communication:** Enable in-app messaging between team members. The system should provide a messaging feature that allows team members to communicate with each other within the app. This can be used to share tips, encourage each other, and coordinate activities.
    - **1.4.2. Social Sharing:**
        - **Description:** Allow users to share trip data and achievements on social media platforms.
        - **Sub-Tasks:**
            - **1.4.2.1. Sharing Options:** Provide options to share trip data and achievements on social media platforms (e.g., Facebook, Twitter). The system should provide options for users to share their trip data and achievements on popular social media platforms. This allows users to easily share their progress with their friends and family.
            - **1.4.2.2. Data Formatting:** Format trip data and achievements for sharing on social media. The system should format the trip data and achievements in a way that is easily readable and shareable on social media. This might involve creating a short summary of the trip or achievement, including the distance traveled, gas saved, CO2e reduced, and money

saved.
        - **1.4.2.3. Social Media Integration:** Integrate with social media APIs to enable sharing. The system should integrate with the relevant social media APIs to allow users to share their data and achievements directly to their social media accounts. This should be done securely and with the user's consent.
        - **1.4.2.4. Friend Invitations:** Allow users to invite friends to join the Sole Power program. The system should provide a feature that allows users to invite their friends to join the Sole Power program. This can be done by sending an invitation email or message to their friends.
- **1.5. Program Information and Resources**
    - **1.5.1. Program Overview:**
        - **Description:** Provide information about the Sole Power program.
        - **Sub-Tasks:**
            - **1.5.1.1. Program Description:** Display a detailed description of the Sole Power program, its goals, and benefits. The system should provide a detailed description of the Sole Power program, including its goals, objectives, and benefits. This should be written in a clear and concise manner, using language that is easy for users to understand.
            - **1.5.1.2. Program Rules:** Display program rules and guidelines. The system should display the rules and guidelines of the Sole Power program. This should include information about eligibility, participation requirements, and how to track trips and earn rewards.
    - **1.5.2. Resources:**
        - **Description:** Link to relevant resources.
        - **Sub-Tasks:**
            - **1.5.2.1. Map Integration:** Integrate with mapping services to display maps, bike paths, and walking trails. The system should integrate with a mapping service, such as Google Maps or Apple Maps, to display maps, bike paths, and walking trails. This allows users to easily find alternative transportation options and plan their routes.
            - **1.5.2.2. Public Transit Information:** Provide links to public transit schedules and routes. The system should provide links to public transit schedules and routes for the local area. This allows users to easily find information about public transportation options and plan their trips.
            - **1.5.2.3. Local Events:** Display information about local events related to sustainable transportation. The system should display information about local events related to sustainable transportation, such as bike races, walking tours, and car-free days. This encourages users to participate in events and learn more about sustainable transportation.
    - **1.5.3. Incentives:**
        - **Description:** Display information about available incentives and rewards.
        - **Sub-Tasks:**
            - **1.5.3.1. Incentive Display:** Display a list of available incentives and rewards. The system should display a list of available incentives and rewards for participating in the Sole Power program. This should include information about the type of incentive, the eligibility requirements, and how to redeem the incentive.
            - **1.5.3.2. Incentive Eligibility:** Provide information on how to qualify for incentives. The system should provide clear information on how to qualify for incentives. This might involve reaching certain mileage goals, participating in events, or referring friends to the program.
            - **1.5.3.3. Incentive Redemption:** Provide instructions on how to redeem incentives. The system should provide instructions on how to redeem incentives. This might involve entering a code, providing proof of participation, or contacting a program administrator.
- **1.6. Notifications and Updates**
    - **1.6.1. Push Notifications:**
        - **Description:** Send push notifications to users about program updates, challenges, events, and incentives.
        - **Sub-Tasks:**
            - **1.6.1.1. Notification Content:** Create and manage notification content. The system should allow administrators to create and manage notification content. This involves writing the notification message, selecting the target audience, and setting the notification type (e.g., alert, reminder, update).
            - **1.6.1.2. Notification Scheduling:** Schedule notifications to be sent at specific times or based on user actions. The system should allow administrators to schedule notifications to be sent at specific times or based on user actions. This allows for targeted notifications that are relevant to the user's needs and interests.
            - **1.6.1.3. Notification Delivery:** Deliver notifications to users' devices. The system should deliver notifications to users' devices using the device's push notification service. This ensures that users receive notifications even when the app is not open.
            - **1.6.1.4. Notification Settings:** Allow users to customize notification settings (e.g., frequency, type). The system should allow users to customize their notification settings. This allows users to control the frequency and type of notifications they receive.
    - **1.6.2. Email Updates:**
        - **Description:** Allow users to subscribe to weekly e-newsletters.
        - **Sub-Tasks:**
            - **1.6.2.1. Subscription Management:** Allow users to subscribe to or unsubscribe from email updates. The system should provide a mechanism for users to subscribe to or unsubscribe from weekly e-newsletters. This allows users to control the frequency and type of communication they receive from the program.
            - **1.6.2.2. Email Content:** Create and manage email content. The system should allow administrators to create and manage the content of the weekly e-newsletters. This involves writing the email message, selecting the target audience,

and setting the email subject line.
        - **1.6.2.3. Email Delivery:** Send email updates to subscribed users. The system should send email updates to subscribed users on a weekly basis. This ensures that users stay informed about program updates, events, and other relevant information.

2. Administrator Role

- **2.1. User Management**
    - **2.1.1. User Account Creation:**
        - **Description:** Create new user accounts for participants.
        - **Sub-Tasks:**
            - **2.1.1.1. User Input:** Provide input fields for username, email address, password, and other relevant user information. The system should provide a form for administrators to enter the user's username, email address, password, and other relevant information. The username should be unique and should not already exist in the database. The password should be hashed for security purposes.
            - **2.1.1.2. Validation:** Validate user input to ensure data integrity. The system should validate the user input to ensure that it is valid and meets the required format. For example, the email address should be validated using a regular expression to ensure a valid format.
            - **2.1.1.3. Role Assignment:** Assign roles and permissions to users. The system should allow administrators to assign roles and permissions to users. This ensures that users have the appropriate access to the system's features and data.
            - **2.1.1.4. Account Creation:** Create a new user account in the database. Once the administrator has entered and validated the user information, the system should create a new user account in the database. This involves storing the user's username, email address, password (hashed), role, and other relevant information.
    - **2.1.2. User Account Management:**
        - **Description:** Edit user profiles and manage user accounts.
        - **Sub-Tasks:**
            - **2.1.2.1. User Search:** Allow administrators to search for users by username, email address, or other criteria. The system should provide a search function that allows administrators to easily find users by username, email address, or other criteria. This allows administrators to quickly access and manage user accounts.
            - **2.1.2.2. User Profile Editing:** Allow administrators to edit user profiles, including username, password, email address, and team affiliation. The system should allow administrators to edit user profiles, including the user's username, password, email address, team affiliation, and other relevant information. This allows administrators to update user information and manage user accounts.
            - **2.1.2.3. Account Deactivation:** Allow administrators to deactivate user accounts. The system should allow administrators to deactivate user accounts. This prevents users from accessing the app's features until their account is reactivated.
            - **2.1.2.4. Account Deletion:** Allow administrators to delete user accounts. The system should allow administrators to delete user accounts. This permanently removes the user's account and all associated data from the system.
    - **2.1.3. Team Management:**
        - **Description:** Manage teams and team members.
        - **Sub-Tasks:**
            - **2.1.3.1. Team Approval:** Approve or deny team creation requests. The system should allow administrators to approve or deny team creation requests. This ensures that only legitimate teams are created and that the program's integrity is maintained.
            - **2.1.3.2. Team Member Addition:** Add users to teams. The system should allow administrators to add users to teams. This allows administrators to manage team membership and ensure that users are assigned to the correct teams.
            - **2.1.3.3. Team Member Removal:** Remove users from teams. The system should allow administrators to remove users from teams. This allows administrators to manage team membership and ensure that users are assigned to the correct teams.
            - **2.1.3.4. Team Editing:** Edit team names and descriptions. The system should allow administrators to edit team names and descriptions. This allows administrators to update team information and ensure that the team information is accurate and up-to-date.
- **2.2. Data Management**
    - **2.2.1. Trip Data Management:**
        - **Description:** View and edit trip data for all participants.
        - **Sub-Tasks:**
            - **2.2.1.1. Trip Data Retrieval:** Retrieve trip data from the database. The system should allow administrators to retrieve trip data from the database. This allows administrators to view and analyze trip data for all participants.
            - **2.2.1.2. Trip Data Display:** Display trip data in a user-friendly format. The system should display trip data in a user-friendly format, such as a table or list. This should include the trip date, time, distance, mode of transportation, and category.
            - **2.2.1.3. Trip Data Editing:** Allow administrators to edit trip data. The system should allow administrators to edit trip data. This allows administrators to correct errors or update trip data if needed.
            - **2.2.1.4. Trip Data Export:** Allow administrators to export trip data in a desired format. The system should allow administrators to export trip data in a desired format, such as CSV or PDF. This allows administrators to easily share

trip data with others or use it for analysis.

- **2.2.2. Leaderboard Management:**
  - **Description:** View and edit leaderboards for the current season and historical data.
  - **Sub-Tasks:**
    - **2.2.2.1. Leaderboard Data Retrieval:** Retrieve leaderboard data from the database. The system should allow administrators to retrieve leaderboard data from the database. This allows administrators to view and manage leaderboards for the current season and historical data.
    - **2.2.2.2. Leaderboard Data Display:** Display leaderboard data in a user-friendly format. The system should display leaderboard data in a user-friendly format, such as a table or list. This should include the user's rank, username, team (if applicable), and the relevant metrics, such as total mileage, gas saved, CO2e reduced, and money saved.
    - **2.2.2.3. Leaderboard Data Editing:** Allow administrators to edit leaderboard data. The system should allow administrators to edit leaderboard data. This allows administrators to correct errors or update leaderboard data if needed.
- **2.2.3. Dashboard Management:**
  - **Description:** View and edit the real-time dashboard for the current challenge.
  - **Sub-Tasks:**
    - **2.2.3.1. Dashboard Data Retrieval:** Retrieve dashboard data from the database. The system should allow administrators to retrieve dashboard data from the database. This allows administrators to view and manage the real-time dashboard for the current challenge.
    - **2.2.3.2. Dashboard Data Display:** Display dashboard data in a user-friendly format. The system should display dashboard data in a user-friendly format, such as a table or list. This should include the total number of trips, total mileage, gas saved, CO2e reduced, and money saved.
    - **2.2.3.3. Dashboard Data Editing:** Allow administrators to edit dashboard data. The system should allow administrators to edit dashboard data. This allows administrators to correct errors or update dashboard data if needed.
    - **2.2.3.4. Dashboard Reset:** Allow administrators to reset the dashboard at the start of each new season. The system should allow administrators to reset the dashboard at the start of each new season. This ensures that the dashboard data is accurate and up-to-date for the new season.

- **2.3. Program Management**
  - **2.3.1. Incentive Management:**
    - **Description:** Create, edit, and manage incentives and rewards for participants.
    - **Sub-Tasks:**
      - **2.3.1.1. Incentive Creation:** Allow administrators to create new incentives. The system should provide a form for administrators to create new incentives. This should include information about the type of incentive, the eligibility requirements, and the redemption process.
      - **2.3.1.2. Incentive Editing:** Allow administrators to edit existing incentives. The system should allow administrators to edit existing incentives. This allows administrators to update incentive information or change the eligibility requirements.
      - **2.3.1.3. Incentive Deletion:** Allow administrators to delete incentives. The system should allow administrators to delete incentives. This allows administrators to remove incentives that are no longer relevant or are no longer being offered.
      - **2.3.1.4. Incentive Assignment:** Assign incentives to participants. The system should allow administrators to assign incentives to participants. This ensures that participants receive the correct incentives based on their eligibility and participation.
  - **2.3.2. Communication Management:**
    - **Description:** Send push notifications and email updates to participants.
    - **Sub-Tasks:**
      - **2.3.2.1. Notification Creation:** Create and manage notification content. The system should allow administrators to create and manage notification content. This involves writing the notification message, selecting the target audience, and setting the notification type (e.g., alert, reminder, update).
      - **2.3.2.2. Notification Scheduling:** Schedule notifications to be sent at specific times or based on user actions. The system should allow administrators to schedule notifications to be sent at specific times or based on user actions. This allows for targeted notifications that are relevant to the user's needs and interests.
      - **2.3.2.3. Notification Delivery:** Deliver notifications to users' devices. The system should deliver notifications to users' devices using the device's push notification service. This ensures that users receive notifications even when the app is not open.
      - **2.3.2.4. Email Content Creation:** Create and manage email content. The system should allow administrators to create and manage the content of the weekly e-newsletters. This involves writing the email message, selecting the target audience, and setting the email subject line.
      - **2.3.2.5. Email Delivery:** Send email updates to subscribed users. The system should send email updates to subscribed users on a weekly basis. This ensures that users stay informed about program updates, events, and other relevant information.
  - **2.3.3. Reporting and Analytics:**
    - **Description:** Generate reports on program participation, trip data, and user engagement.
    - **Sub-Tasks:**
      - **2.3.3.1. Report Generation:** Generate reports on program participation, trip data, and user engagement. The system should allow administrators to generate reports on program participation, trip data, and user engagement. This allows

administrators to track the program's progress and identify areas for improvement.
        - **2.3.3.2. Data Visualization:** Visualize data in charts, graphs, and other formats. The system should allow administrators to visualize data in charts, graphs, and other formats. This helps administrators to easily understand and interpret the data.
        - **2.3.3.3. Data Analysis:** Analyze data to identify trends and areas for improvement. The system should allow administrators to analyze data to identify trends and areas for improvement. This allows administrators to make informed decisions about the program and its future development.
- **2.4. System Administration**
    - **2.4.1. App Updates:**
        - **Description:** Manage app updates and releases.
        - **Sub-Tasks:**
            - **2.4.1.1. Update Development:** Develop and test app updates. The system should allow administrators to develop and test app updates. This involves fixing bugs, adding new features, and improving the app's performance.
            - **2.4.1.2. Update Deployment:** Deploy app updates to app stores. The system should allow administrators to deploy app updates to app stores. This involves submitting the app update to the relevant app store (e.g., Apple App Store, Google Play Store) for review and approval.
            - **2.4.1.3. Update Rollout:** Manage the rollout of app updates to users. The system should allow administrators to manage the rollout of app updates to users. This involves controlling the release schedule and ensuring that users receive the updates in a timely manner.
    - **2.4.2. Security Management:**
        - **Description:** Implement and maintain security measures to protect user data.
        - **Sub-Tasks:**
            - **2.4.2.1. Access Control:** Implement access control mechanisms to restrict unauthorized access to user data. The system should implement access control mechanisms to restrict unauthorized access to user data. This involves using authentication and authorization mechanisms to ensure that only authorized users can access sensitive data.
            - **2.4.2.2. Data Encryption:** Encrypt sensitive user data at rest and in transit. The system should encrypt sensitive user data at rest and in transit. This involves using encryption algorithms to protect user data from unauthorized access.
            - **2.4.2.3. Security Audits:** Conduct regular security audits to identify and address vulnerabilities. The system should undergo regular security audits to identify and address vulnerabilities. This involves using security scanning tools and techniques to identify potential security risks and implement appropriate security measures.
    - **2.4.3. System Maintenance:**
        - **Description:** Perform regular system maintenance and backups.
        - **Sub-Tasks:**
            - **2.4.3.1. System Monitoring:** Monitor system performance and identify potential issues. The system should be monitored regularly to ensure that it is performing as expected. This involves monitoring system performance metrics, such as CPU usage, memory usage, and network traffic.
            - **2.4.3.2. System Backups:** Perform regular backups of system data. The system should be backed up regularly to prevent data loss in case of a system failure. This involves creating copies of the system's data and storing them in a secure location.
            - **2.4.3.3. System Updates:** Apply system updates and patches. The system should be updated regularly with the latest security patches and updates. This helps to protect the system from vulnerabilities and ensure that it is running smoothly.
            - **2.4.3.4. Issue Resolution:** Troubleshoot and resolve technical issues. The system should be monitored for technical issues, and any issues should be resolved promptly. This involves identifying the cause of the issue, implementing a solution, and testing the solution to ensure that it is effective.

3. Additional Roles (Optional)

- **3.1. Sponsor Role:**
    - **Description:** View program data and analytics. *